(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **01156824 A**

(43) Date of publication of application: **20.06.89**

(51) Int. Cl **G06F 9/30**
G06F 7/00
G06F 9/32
G06F 15/72

(21) Application number: **62314063**

(22) Date of filing: **14.12.87**

(71) Applicant: **HITACHI LTD HITACHI MICRO COMPUT ENG LTD SAKAMURA TAKESHI**

(72) Inventor: **SAKAMURA TAKESHI**
KAWASAKI IKUYA
HASEGAWA ATSUSHI
IWASAKI KAZUHIKO

(54) **MICROPROCESSOR**

(57) Abstract:

PURPOSE: To secure the program flexibility and to facilitate the development of a program for the graphic processing, for example, by storing the information obtained by execution of a 1st instruction to an information holding means and controlling an instruction executing means based on said stored information when a 2nd instruction is carried out.

CONSTITUTION: The type of an arithmetic operation is defined as one of operands. In other words, a desired arithmetic operation is carried out by an instruction that is added the operand information designating the type of the arithmetic operation to the outside or the inside of an operation designating part containing a common operation code showing an arithmetic operation (wide sense). Thus the contents of an operand are set previously to designate the type of the arithmetic operation based on the executing result of a certain instruction. Then the arithmetic operation is carried out based on the operand contents by the next instruction. Thus the type of the arithmetic operation can be changed into a dynamic one in a program. In such a way, the flexibility is secured to a program and the development of a program for graphic processing, for example, is facilitated.

⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　平1－156824

⑤Int.Cl.⁴		識別記号	庁内整理番号	⑬公開　平成1年(1989)6月20日
G 06 F	9/30	340	A－7361－5B	
	7/00		B－7313－5B	
	9/32	350	B－7361－5B	※審査請求　未請求　発明の数 2　(全16頁)

⑭発明の名称　マイクロプロセッサ

⑳特　願　昭62－314063
㉒出　願　昭62(1987)12月14日

㉑発明者　坂村　健　東京都港区白金台3－12－30－105
㉑発明者　川崎　都也　東京都小平市上水本町1450番地　株式会社日立製作所武蔵工場内
㉑出願人　株式会社日立製作所　東京都千代田区神田駿河台4丁目6番地
㉑出願人　日立マイクロコンピュータエンジニアリング株式会社　東京都小平市上水本町1479番地
㉑出願人　坂村　健　東京都港区白金台3－12－30－105
㉔代理人　弁理士　小川　勝男　外1名

最終頁に続く

明　細　書

1. 発明の名称

マイクロプロセッサ

2. 特許請求の範囲

1. 命令解読手段と、
命令実行手段と、
情報保持手段とを含み、
第1の命令の実行によって得られた情報を上記情報保持手段に格納する第1のステップと、第2の命令の実行の際上記情報に基づいて上記命令実行手段が制御される第2のステップを含むことを特徴とするマイクロプロセッサ。
2. 上記命令実行手段は論理演算ユニットを含み、上記情報に基づいて、上記論理演算ユニットの演算機能が制御されることを特徴とする特許請求の範囲第1項記載のマイクロプロセッサ。
3. 上記情報保持手段は汎用レジスタであることを特徴とする特許請求の範囲第2項記載のマイクロプロセッサ。
4. 命令解読手段と、
命令実行手段とを含み、
上記命令解読手段によって解読される命令のオペコード内には、書替え可能な情報保持手段に記憶された情報に基づいて上記命令実行手段の少なくとも一部の制御を実行させる情報を含むことを特徴とするマイクロプロセッサ。
5. 上記情報が記憶される上記情報保持手段内の番地を指定するための番地情報は、上記オペコード内に含まれていることを特徴とする特許請求の範囲第4項記載のマイクロプロセッサ。
6. 上記情報が記憶される上記情報保持手段内の番地を指定するための番地情報は、オペランド領域に含まれていることを特徴とする特許請求の範囲第4項記載のマイクロプロセッサ。
7. 上記書替え可能な情報保持手段とは、マイクロプロセッサ内のレジスタであることを特徴とする特許請求の範囲第4項記載のマイクロプロセッサ。
8. 上記命令実行手段は論理演算ユニットを含み、上記レジスタの内容に基づいて上記論理演算ユ

ニットの演算機能が制御されることを特徴とする特許請求の範囲第7項記載のマイクロプロセッサ。

9. 上記レジスタの内容に基づいて上記論理演算ユニットの演算の種類が選択されることを特徴とする特許請求の範囲第8項記載のマイクロプロセッサ。

10. 上記命令解読手段によって解読される命令は、メモリ内の任意のビットから任意のビットまでの領域のデータの取り扱いに関する命令であることを特徴とする特許請求の範囲第9項記載のマイクロプロセッサ。

3. 発明の詳細な説明

〔産業上の利用分野〕

この発明は、データ処理技術さらにはプログラム制御方式のシステムにおける命令体系に適用して特に有効な技術に関し、例えばビットフィールドと呼ばれるデータの取り扱いに関する命令を有するマイクロプロセッサに利用して有効な技術に関する。

例えば、コンピュータ・グラフィックのような分野においては、ビットフィールド内のデータに対し論理演算を施して、いわゆる塗りつぶしやすかしなどの描画処理を行なわせる場合、画面を見ながら演算の種類をダイナミックに決定することができればプログラムの開発が容易となる。

しかるに、演算の種類によって命令が決っている従来のマイクロプロセッサでは、演算処理の内容を変えるにはプログラムの中の演算命令を書き換えなくてはならず、プログラムに柔軟性がないという不都合があった。

また、ある命令の実行によって得られた結果に基づいて次の命令又は演算の種類を決定するためには、プログラム上、次に実行する可能性のある命令又は演算を羅列しておかなければならない。すなわち、上記ある命令の実行によって得られた結果に基づいて上記羅列された命令等の1つを選択する様なプログラムを作成しなければならない。従ってプログラムの柔軟性がないだけでなく、命令を選択する等の処理を必要とするから、一連の命令を実行する際の高速動作が制限されてしまう。

〔従来の技術〕

従来、マイクロプロセッサには、加算、減算、乗算、除算、比較などの算術演算命令の他、論理積（ＡＮＤ）、論理和（ＯＲ）、排他的論理和（ＸＯＲ）など種々の論理演算命令が備えられている。例えば（株）日立製作所、１９８２年９月発行、「日立マイクロコンピュータ、ＳＥＭＩＣＯＮＤＵＣＴＥＲ ＤＡＴＡ ＢｏｏＫ、８／１６ビットマイクロコンピュータ」ｐ９１４～ｐ９１９，ｐ９４５～ｐ９５２等に記載されている。

〔発明が解決しようとする問題点〕

従来のマイクロプロセッサにおける命令体系では、演算の種類は命令（オペレーションコード）で指定するようにされていた。つまり、各演算ごとに命令が用意されており、演算の種類はプログラム上において固定されてしまい、データのように変更することはできなかった。従って、プログラムがＲＯＭ（リード・オンリ・メモリ）内に格納された場合、演算を変更することは不可能であった。

この発明の目的は、マイクロコンピュータ・システムにおけるプログラムに柔軟性を持たせ、例えばグラフィック処理用のプログラムの開発が容易に行なえるような演算命令に関する命令形式を提供することにある。

この発明の前記ならびにその他の目的と新規な特徴については、本明細書の記述および添附図面から明らかになるであろう。

〔問題を解決するための手段〕

本願において開示される発明のうち代表的なものの概要を説明すれば、下記のとおりである。

すなわち、演算の種類をオペランドの一つとして与える、つまり演算（広義）という共通のオペレーションコードの入ったオペレーション指定部の外又は中に、演算の種類を指定するオペランド情報を付加した命令によって所望の演算を実行させるようにするものである。

〔作用〕

上記した手段によれば、　る命令の実行結果に

基づいて、演算の種類を指定するオペランドの内容を設定しておき、次の命令で上記オペランドの内容に従って演算できるから、プログラムの中で演算の種類をダイナミックに変えることができる。従ってプログラムに柔軟性を持たせ、例えばグラフィック処理用のプログラムの開発が容易に行なえるようにするという上記目的を達成することができる。

以下、本発明を一例として、ビットフィールドと呼ばれるメモリ内の任意のビットから任意のビットまでのデータの取り扱いに関する命令（以下ビットフィールド命令と称する）に適用した場合の一実施例を説明する。

〔実施例〕

ビットフィールド命令は、第1図及び第2図に示すようにベースアドレスＢＡとこのベースアドレスからのオフセットＯｆｆおよびフィールド長さ（ビット数）を示すフィールド幅ＷＤの3つの値をオペランドとして与えることによりメモリ内の所望のフィールドを指定し、そのフィールドのデータに対してアンド（ＡＮＤ）やオア（ＯＲ）などの論理演算処理を行なうものでる。なお、このようなビットフィールド命令は、例えばモトローラ社製ＭＣ６８０２０のようなマイクロプロセッサで既に用意されている。このビットフィールド命令は、オペレーションコードの後のオペランドによってベースアドレスＢＡやオフセットｏｆｆおよびフィールド幅ＷＤが与えられていた。

この実施例では演算の種類もオペランドで指定するものである。オペランドによる演算の指定の具体的な方法として、例えば第9図(A)の実施例では、レジスタ番号を有するレジスタ直接アドレッシング方式を用いた。すなわち、予め所定のレジスタＲＳ内に演算の種類を示すコードを入れておき、オペランドにはそのコードの入ったレジスタ番号とアドレッシングモードを入れておくものである。第9図(B)の実施例では、オペレーションコード内に、レジスタＲ５の内容に基づいて演算の種類を定める旨の情報が付加されている。第9図(A)又は、第9図(B)に示す命令を実行する場合には、

演算の種類を示すコードは、予めＭＯＶＥ命令等によりメモリ内からデータとして読み出して所定のレジスタＲ５に入れておく。

また、同様にベースアドレスＢＡ，オフセットＯｆｆ及びフィールド幅ＷＤもそれぞれ所定のレジスタ内に入っている値を使って命令を実行するようにされている。

第9図(A)又は第9図(B)に示す命令は、例えば、ビットフィールド命令であり、あるビットフィールド（ソース側）のデータと他のビットフィールド（デスティネーション側）のデータとの論理をとってそれをデスティネーション側のビットフィールドへ入れるというビットフィールド間演算命令である。この命令を実行するためには、ソース側のビットフィールドを特定するベースアドレスＢＡｓ，オフセットＯｆｆｓ及びフィールド幅ＷＤｓをそれぞれ入れるレジスタと、デスティネーション側ビットフィールドを特定するベースアドレスＢＡｄ，オフセットＯｆｆｄ及びフィールド幅ＷＤｄをそれぞれ入れるレジスタと、演算の種類を特定するコードを入れるレジスタとが必要である。ただし、上記のような２つのビットフィールドの論理をとる命令ではフィールド幅ＷＤは必然的に同一であるので、レジスタは共用させることができる。

第1表には、上記ビットフィールド間演算命令において使用されるレジスタとその中に格納されるデータとの関係の一例が示されている。

第 1 表

Ｒ0	ソース側ビットフィールドのＢＡ
Ｒ1	ソース側ビットフィールドのＯｆｆ
Ｒ2	フィールド幅（ＷＤ）
Ｒ3	デスティネーション側ビットフィールドのＢＡ
Ｒ4	デスティネーション側ビットフィールドのＯｆｆ
Ｒ5	演算の種類

同図における符号ＢＡはベースアドレス，Ｏｆｆはオフセットを示す。

また、第2表には、上記レジスタＲ５によって指定される演算の種類の一覧表が示されている。

第 2 表

№	演算の種類	内　　容
1	True	1→dest
2	False	0→dest
3	NotDest	$\overline{dest}$→dest
4	Dest	dest→dest
5	NotSrc	$\overline{src}$→dest
6	Src	src→dest
7	AND	dest. and. src→dest
8	Or	dest. or. src→dest
9	Xor	dest. xor. src→dest
10	NotAnd	$\overline{dest}$. and. src→dest
11	NotOr	$\overline{dest}$. or. src→dest
12	AndNot	dest. and. $\overline{src}$→dest
13	OrNot	dest. or. $\overline{src}$→dest
14	NotAndNot	$\overline{dest}$. and. $\overline{src}$→dest
15	NotOrNot	$\overline{dest}$. or. $\overline{src}$→dest
16	NotXor	$\overline{dest}$. xor. src→dest

同表において、Ｔｒｕｅで示される演算はデスティネーション側ビットフィールドの全ビットを"１"にする操作を、Ｆａｌｓｅで示される演算はデスティネーション側ビットフィールドの全ビットを"０"にする操作を意味する。また、Ｎｏｔ－Ｄｅｓｔで示される演算はデスティネーション側ビットフィールド内の全ビットのデータを反転して元のビットフィールドに入れる操作を、Ｄｅｓｔで示される演算はデスティネーション側ビットフィールド内のデータをそのまま元のビットフィールドに戻す操作を、Ｎｏｔで示される演算はソース側ビットフィールド内の全ビットのデータを反転してデスティネーション側ビットフィールドに入れる操作を、そして、Ｓｒｃで示される演算はソース側ビットフィールド内のデータをデスティネーション側ビットフィールドに入れる操作を意味する。

さらに、ＡＮＤで示される演算は、ソース側とデスティネーション側のビットフィールド内のデータの論理積をとってデスティネーション側のビットフィールドに入れる操作を、Ｏｒで示される演算は、ソース側とデスティネーション側のビットフィールド内のデータの論理和をとってデスティネーション側のビットフィールドに入れる操作を、Ｘｏｒで示される演算は、ソース側とデスティネーション側のビットフィールド内のデータの排他的論理和をとってデスティネーション側のビットフィールドに入れる操作を、ＮｏｔＡｎｄで示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内データとの論理積をとってデスティネーション側のビットフィールドに入れる操作を、ＮｏｔＯｒで示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内データとの論理和をとってデスティネーション側のビットフィールドに入れる操作を、ＡｎｄＮｏｔで示される演算は、デスティネーション側のビットフィールド内のデータとソース側ビットフィールド内データの反転値との論理積をとってデスティネーション側のビットフィールドに入れる操作を、ＯｒＮｏｔで示される演算は、デスティネーション側のビットフィールド内のデータとソース側ビットフィールド内データの反転値との論理和をとってデスティネーション側のビットフィールドに入れる操作を、ＮｏｔＡｎｄＮｏｔで示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内データ反転値との論理積をとってデスティネーション側のビットフィールドに入れる操作を、ＮｏｔＯｒＮｏｔで示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内データの反転値との論理和をとってデスティネーション側のビットフィールドに入れる操作を、ＮｏｔＸｏｒで示される演算は、デスティネーション側のビットフィールド内のデータの反転値とソース側ビットフィールド内データとの排他的論理和をとってデスティネーション側のビットフィールドに入れる操作をそれぞれ意味する。

上記各種演算は例えばレジスタR5の下位4ビットによって識別させることができる。

上記のようなビットフィールド間演算命令を使用すると、演算の種類がオペランドの一つとして与えられるため、メモリ内のデータを変更するかメモリからロードするデータを変えるだけでプログラム実行中に演算の種類をダイナミックに変更することができる。ただし、この実施例のビットフィールド間演算命令を実行する前に、予めオペランドとして与えられるベースアドレスやオフセットおよび演算の種類を示すコードを所定のレジスタ（R_0～R_n）に入れておいてやる必要がある。

第3表に、上記ビットフィールド間演算命令（BVMAPと略す）を使用したグラフィック表示用のプログラムの一例が示されている。

第3表

```
LOOP    MOVE  (R10)+,R0
        MOVE  (R11)+,R1
        MOVE  (R12)+,R2
        MOVE  (R13)+,R3
        MOVE  (R14)+,R4
        MOVE  (R15)+,R5
        BVMAP
        SUB   LINE,-1
        BNE   LOOP
```

上記プログラムは、ラインごとにポストインクリメントよってレジスタR0～R5の内容を変えながらBVMAPで示されるビットフィールド間演算命令を繰返し実行することを処理の内容としている。例えばMOVE(R10)+,R0は、レジスタR0の内容を更新してレジスタR0に格納する命令である。またSUB LINE,-1は、総ライン数から1を引く命令である。

従って、上記プログラムにおいてレジスタR5に格納される演算種類を例えば一回ごとに変えながら繰返し実行してやれば、表示画面上において一行ごとに演算内容の異なるビットフィールド処理を行なった結果が画像として表示されるようになる。

上記プログラムでは、ラインごとにレジスタR5の内容を変えることにより演算の種類を変更しているが、本発明はこの実施例に限定されるものではない。例えば上記プログラムの前に実行されるプログラムによって得られた結果をレジスタR5に格納しておき、レジスタR5の内容を更新せずに上記プログラムを実行することもできる。これにより、プログラム実行中に演算の種類をダイナミックに変更することができる。

第7図にはビットフィールド間演算命令BVMAPのフローチャートが示されている。ステップS1において、レジスタR0，R1及びR2の内容を用いてソース側ビットフィールドがフエッチされる。ステップS2においてレジスタR2，R3及びR4の内容を用いてデスティネーション側ビットフィールドがフエッチされる。ステップS3において、レジスタR5の内容を用いて演算が実行される。ステップS4において終了条件を判定し、終了条件が一致すれば命令を終了し、不一致ならば上記ステップ1に戻る。1度にフエッチできるデータのビット数はマイクロプロセッサのデータバス長によって定まる。従って、ビットフィールドの全てをフエッチし、これに基づいて演算を行うためには上記ステップS1～S3を複数回繰り返すことが必要な場合がある。

なお、上記実施例では、ビットフィールド命令として2つのビットフィールドのデータ同士の操作に関する命令を例に挙げて説明したが、グラフィック処理に適したビットフィールド命令としては、その他に例えばベースアドレスとオフセット及びフィールド幅で指定されたビットフィールドに対して任意のレジスタのビットパターンを繰返し格納させるような命令が考えられる。この命令を使用すると画面上の任意の領域を任意のパターン（模様を構成する基本図形）で塗めて行くような一種の塗りつぶし処理が容易に行なえるようになる。

第3図には、上記実施例のビットフィールド命

令を有する命令体系によって動作するマイクロプロセッサのハードウエア構成の一例が示されている。

この実施例のマイクロプロセッサは、マイクロプログラム制御方式の制御部を備えている。すなわち、マイクロプロセッサを構成するＬＳＩチップＩ内には、マイクロプログラムが格納されたマイクロＲＯＭ（リード・オンリ・メモリ）２が設けられている。マイクロＲＯＭ２は、マイクロアドレス発生回路５によってアクセスされ、マイクロプログラムを構成するマイクロ命令を順次出力する。

マイクロアドレス発生回路５は、命令レジスタ３にフエッチされたマイクロ命令のコードを、命令デコーダ４でデコードした信号が供給される。マイクロアドレス発生回路５はこの信号に基づいて対応するマイクロアドレスを形成し、マイクロＲＯＭ２に供給する。これによって、そのマイクロ命令を実行する一連のマイクロ命令群の最初の命令が読み出される。このマイクロ命令コードによって、各種レジスタやデータバッファ、演算論理ユニット等からなる実行ユニット６等に対する制御信号が形成される。この実行ユニット６内に上記実施例で使用された汎用レジスタＲ０～Ｒ15が含まれている。

マクロ命令に対応する一連のマイクロ命令群のうち２番目以降のマイクロ命令の読出しは、直前に読み出されたマイクロ命令のネクストアドレスフィールドのコードがマイクロＲＯＭ２に供給されることより、直前のマイクロ命令内のネクストアドレスとマイクロアドレス発生回路５からのアドレスとに基づいて行われる。このようにして、一連のマイクロ命令が読み出されて形成された制御信号によって実行ユニット６が制御され、マクロ命令が実行される。

この実施例では、特に制限されないが、バッファ記憶方式が採用されており、マイクロプロセッサＬＳＩ内にキャッシュメモリ７が設けられ、外部メモリ８内でのデータのうちアクセス頻度の高いプログラムデータがキャッシュメモリ７内に登録される。これによって、プログラムの取込みが高速化される。

なお、上記実施例では、一例としてグラフィック処理に適したビットフィールド命令に適用したものについて説明したが、それ以外の演算命令に適用することができる。

また、上記実施例では、オペランドで指定する演算の種類がアンド（ＡＮＤ）やオア（ＯＲ）などの論理演算に限定されているが、算術演算を行なう命令についても同様にオペレーションコードを同一にし、かつ演算の種類をオペランドで指定するようにしてもよい。

第４図は、第３図に示す実行ユニット６の内部ブロック図を示している。

第４図の実行ユニットにおいて、回路符号ＣＢＳで示されているのは、オフセット値やフィールド幅等の拡張データをラッチするためのレジスタ、ＤＯＲはメモリへストアするデータをラッチするためのデータ・アウトプット・レジスタ、ＤＩＲはメモリから読み出されたデータをラッチするデータ・インプット・レジスタ、ＡＬＮは入出力されるデータを整列させるアライナで、このアライナＡＬＮはデータＩ／Ｏインタフェース（図示省略）を介して外部のデータバスに接続される。

また、回路符号ＢＳＦで示されるのは、３２ビットずつ同時に入力された６４ビットデータから任意の３２ビットを抽出するためのバレルシフタで、このバレルシフタＢＳＦは、０のような定数を直接入力できるように構成されている。ＢＣＮＴはバレルシフタＢＳＦに対して抽出する位置を指定するバレルシフタ・カウンタ、ＢＳＦＯはバレルシフタＢＳＦの出力をラッチするレジスタである。また、ＦＢはデータを入力させることにより上位２７ビットをマスクして出力する等の働きをするファンクション・ブロック、ＦＢＯはファンクション・ブロックＦＢの出力をラッチするレジスタである。

さらに、回路符号ＡＵで示されるのは実効アドレスを計算するためのアドレス演算ユニットで、このアドレス演算ユニットＡＵは、０のような定

数を直接入力できるように構成されている。ＡＵＯはこのアドレス演算ユニットＡＵの出力をラッチするレジスタ、ＳＰＴはアドレス演算ユニットＡＵで演算される前のデータのシフトを行なうシフタ、ＡＯＴは演算結果の入っている上記レジスタＡＵＯの値を後述のテンポラリレジスタＤＴＥ０～ＤＴＥ３へ移す際に一時的に保持するラッチ回路、ＡＯＲは同じくレジスタＡＵＯのアドレス値を外部へ出力する際に一時的に保持するアドレス・アウトプット・レジスタで、このレジスタＡＯＲはアドレスＩ／Ｏインタフェース（図示省略）を介して外部のアドレスバスに接続される。

一方、回路符号ＡＬＵで示されているのは、加算、減算等基本的な算術演算や論理演算を行なう演算論理ユニット、ＡＬＵＯは演算論理ユニットＡＬＵでの演算結果をラッチするレジスタ、また、ＤＴＥ０～ＤＴＥ３で示されるのは、テンポラリ値をラッチする外部から見えない（ユーザーに開放されていない）レジスタ群、R_0，R_1，……R_{15}はユーザに開放されている汎用レジスタ群である。

本発明によれば上記演算論理ユニットＡＬＵ等は汎用レジスタの内容、例えばレジスタＲ５の内容によって制御可能とされる。

第５図は、第４図に示す演算論理ユニットＡＬＵ等及びこれらを制御するためのレジスタＲ５との関係を示している。

汎用レジスタＲ５の内容は、特に限定されないが一旦他のレジスタＩＮＦＲにストアされ、このレジスタＩＮＦＲから制御信号Ｉ１～Ｉ５が出力される。レジスタＩＮＦＲは第４図において省略されているが、上記テンポラリレジスタＤＴＥＯ等と同様のレジスタとされ、実行ユニット内に設けられる。制御信号Ｉ１は、ＢＢバス上のデータ又はオールゼロ(0)のデータの一方を選択するための信号である。制御信号Ｉ２は、インバータ回路ＩＮＶの動作を制御する信号であり、入力信号を反転して出力するか、反転せずに出力するかの選択をする信号である。制御信号Ｉ３は演算論理ユニットＡＬＵの演算機能の選択をする信号である。演算論理ユニットＡＬＵは、論理積（ＡＮＤ）、論理和（ＯＲ）又は排他的論理和（ＸＯＲ）等の演算機能を有しており、いずれかの機能が制御信号Ｉ３によって選択されることになる。制御信号Ｉ５はレジスタＡＬＵＯにラッチされたデータをＢＣバスに送出するか又は演算論理ユニットＡＬＵの入力側に帰還するかの選択を行う信号である。制御信号Ｉ４は、上記入力側に帰還されたデータ又はオールゼロ(0)の一方を選択するための信号である。制御信号Ｉ１によって選択されたデータはインバータ回路ＩＮＶを介して演算論理ユニットＡＬＵの一方の入力データとされ、制御信号Ｉ４によって選択されたデータは演算論理ユニットＡＬＵの他方の入力データとされる。この演算論理ユニットＡＬＵの動作は２段階に分けられる。例

第　４　表

No.	Ｉ１による選択	Ｉ２による選択	Ｉ３による選択
1	0	非反転	ＯＲ
2	0	非反転	ＯＲ
3	0	非反転	ＯＲ
4	0	非反転	ＯＲ
5	src	反　転	ＯＲ
6	src	非反転	ＯＲ
7	src	非反転	ＯＲ
8	src	非反転	ＯＲ
9	src	非反転	ＯＲ
10	src	非反転	ＯＲ
11	src	非反転	ＯＲ
12	src	反　転	ＯＲ
13	src	反　転	ＯＲ
14	src	反　転	ＯＲ
15	src	反　転	ＯＲ
16	src	非反転	ＯＲ

5　表

番	I1による選択	I2による選択	I3による選択
1	0	反　転	OR
2	0	非反転	OR
3	dest	反　転	OR
4	dest	非反転	OR
5	0	非反転	OR
6	0	非反転	OR
7	dest	非反転	AND
8	dest	非反転	OR
9	dest	非反転	XOR
10	dest	反　転	AND
11	dest	反　転	OR
12	dest	非反転	AND
13	dest	非反転	OR
14	dest	反　転	AND
15	dest	反　転	OR
16	dest	反　転	XOR

えば、前記第2表に記載した各演算を行う場合において、第1段階の動作状態が第6図(A)に示され、第2段階の動作状態が第6図(B)に示されている。

上記第4表は、上記第6図(A)に示す動作状態をさらに詳細に説明するための表であり、第5表は上記第6図(B)に示す動作状態をさらに詳細に説明するための表である。例えば第2表に示す第1番目の演算1→destを実行する場合には、まず、第6図(A)において、第4表の第1番目に記載された制御が行なわれる。すなわち、制御信号I1によってオールゼロ(0)が選択され、これがインバータ回路INDで反転されずに演算論理ユニットALUの一方の入力データとされる。また演算論理ユニットALUの他方の入力データはオールゼロ(0)とされる。この演算論理ユニットALUの演算機能は、制御信号I3によって論理和（OR）とされるから、その演算結果はオールゼロ(0)とされる。次に第6図(B)において、第5表の第1番目に記載された制御が行なわれる。すなわち制御信号I1によってオールゼロ(0)が選択され、これがインバータ回路INVで反転され、オールワン(1)にされて演算論理ユニットALUの一方の入力データとされる。また演算論理ユニットALUの他方の入力データは上記演算結果（オールゼロ）とされる。演算論理ユニットALUの演算機能は、制御信号I3によって論理和（OR）とされるから、その演算結果はオールワン(1)とされる。このデータをデスティネーション側ビットフィールドにストアすることによって、上記演算1→destの実行が終了する。

また、第10番目の演算$\overline{\text{dest}}$. AND. src→destを実行する場合には、まず、第6図(A)において、第4表の第10番目に記載された制御が行なわれる。すなわち、制御信号I1によって、BB上のデータ（この場合ソース側ビットフィールドの値）srcが選択され、これがインバータ回路INDで反転されずに演算論理ユニットALUの一方の入力データとされる。また演算論理ユニットALUの他方の入力データはオールゼロ(0)とされる。

この演算論理ユニットALUの演算機能は、制御信号I3によって論理和（OR）とされるから、その演算結果はソース側ビットフィールドの値（src）とされる。次に第6図(B)において、第5表の第10番目に記載された制御が行なわれる。すなわち制御信号I1によってBB BUS上のデータ（この場合デスティネーション側ビットフィールドの値）destが選択され、これがインバータ回路INVで反転され（$\overline{\text{dest}}$）演算論理ユニットALUの一方の入力データとされる。また演算論理ユニットALUの他方の入力データは上記演算結果（src）とされる。演算論理ユニットALUの演算機能は、制御信号I3によって論理積（AND）とされるから、その演算結果は$\overline{\text{dest}}$. AND. src とされる。このデータをデスティネーション側ビットフィールドにストアすることによって、上記演算$\overline{\text{dest}}$. AND. src→destの実行が終了する。

上記実施例では、レジスタR5の内容によって、演算論理ユニットALU等が直接制御されている

が、本発明の適用に際して上記実施例に限定されるものではない。すなわち汎用レジスタR5の内容によって演算論理ユニットALU等が間接的に制御されるものであってもよい。例えば、レジスタR5の内容を第3図に示す命令デコーダ4等に供給することにより、マイクロROM2から上記制御信号I1～I5等を得ることもできる。また上記実施例では、制御信号I1はBBバス上のデータを選択するか否かの選択信号とされているが、これに限定されない。例えば、BBバス上にデータを供給するための供給源等を、この制御信号I1で制御することもできる。なお、BBバス上への、ソース側ビットフィールドの値src又はデスティネーション側ビットフィールドの値destの送出は、パレルシフタBSFの出力ラッチレジスタBSFOからなされる。

第8図は、第7図に示すフローチャートにおけるステップS1をさらに詳細に説明するための実行手順が示されている。ステップS1は、ステップS01からS12によって構成される。なお、第4図に示す汎用レジスタ群R0～R15の中で、符号Ra，Rb，Rx，Ryが付記されているレジスタはそれぞれソース・ベースアドレス、デスティネーション・アドレス、オフセット値・アドレス及びビットフィールド幅を入れるために使用されているレジスタであることを示している。このRa，Rb，Rx，Ryは、汎用レジスタR0～R15の任意のレジスタ番号をそれぞれの使用目的のために指定して使用することができる。

第1のステップS01では、レジスタRx内の値すなわちオフセット値の入っている位置を示すアドレスがバスBAまたはBBを介して、また定数入力機能によって0がアドレス演算ユニットAUにそれぞれ入力され、その加算結果がレジスタAUOに格納される。

第2のステップS02では、ステップS01でレジスタAUOに格納されたアドレス値（オフセット値アドレス）がレジスタAORへ転送されるとともに、I／Oインタフェースに対して外部データバス上のデータをフェッチするようにリクエスト指令が与えられる。これによって、レジスタAOR内のアドレス値がI／Oインタフェースを介して外部アドレスバス上に出力され、外部のメモリがアクセスされてその内容がデータバス上に出力される。そして、メモリから読み出されたデータすなわちオフセット値OffがI／Oインタフェースによりフェッチされる。

第3のステップS03では、I／Oインタフェースより出力される信号に基づいてフェッチしたデータが確定しているか確認される。これによって、データはデータ・インプット・レジスタDIRに取り込まれる。これとともに、ステップS03では、レジスタRaの値すなわちソース・ベースアドレスBADがバスBAまたはBBを介して、また定数入力機能によって0が、アドレス演算ユニットAUにそれぞれ入力され、その加算結果がレジスタAUOに格納される。

それから、第4のステップS04で、上記レジスタAUOの値（ソース・ベースアドレス）がレジスタAORへ転送され、更にバスBCを介してテンポラリレジスタDTE0に転送される。

なお、他の処理との関係でアドレス演算ユニットAUの演算結果がレジスタAUOからAORへ転送されるとき、自動的にレジスタAOTにも転送されるようにされている。ここでは、レジスタAOTへの転送は特別の意味も持っていない。これと並行して、アドレス演算ユニットAUには、定数入力機能により0が入力されるとともに、バスBBを介してデータ・インプット・レジスタDIRの内容（オフセット値）が符号拡張されて入力され、演算結果がレジスタAUOに格納される。さらに、レジスタRyからファンクションブロックFBに対して保持値すなわちビットフィールド幅WBがバスBAを介して供給され、ファンクションブロックFBで下位5ビットを除く上位27ビットがマスクされ、結果がレジスタFBOに格納される。ビットフィールド幅の下位5ビットのみ抽出することは、数学的に表現するとビットフィールド幅を数「32」で割った余りを求めることと同様である。以下、このビットフィールド

幅の下位5ビットを端数WD・と記す。ここで端数WD・を求めるのは、後のステップS09でのバウンダリ渡りの判定に使用するためである。

次に、ステップS06でレジスタAUOの値すなわちオフセット値Offをバスを介してテンポラリレジスタDTE1へ転送する。これとともに、アドレス演算ユニットAUに対してレジスタAUOの値（オフセット値）とテンポラリレジスタDTE0の値すなわちソース・ベースアドレスBADをシフタSFTで上位側へ3ビットシフトした値とが供給され、その加算結果がレジスタAUOに格納される。ソース・ベースアドレスBADを上位側へ3ビットシフトするのは、メモリ空間をバイト単位で区切って指示できるようにされたベースアドレスBADを、ビット単位でメモリ空間内での位置を指示できるように拡張するためである。従って、このときレジスタAUOに入っているのは、求めるビットフィールドのアドレス0番地からのビット数で表した距離である。この距離をLと記す。

第6のステップS06では、上記レジスタAUOの値すなわちベースアドレスを上位側へ3ビットシフトした値にオフセット値Offを加えたものを、レジスタAOTに転送する。一方、アドレス演算ユニットAUには、バスBAを介してテンポラリレジスタDTE0からソース・ベースアドレスBADが入力され、またバスBBを介してテンポラリレジスタDTE1から転送されたオフセット値Offをシフタ SFTで下位側へ3ビットシフトした値が入力されて加算され、その加算結果の下位2ビットをマスクした値がレジスタAUOに格納される。これとともに、レジスタFBO内の値WD・すなわちビットフィールド幅の下位5ビットがバスBCを介してテンポラリレジスタDTE3に転送される。

上記の場合、アドレス演算ユニットAUによって、ベースアドレスに、オフセット値を下位側へ3ビットだけシフトした値を加算しているのは、対象となるビットフィールドの先頭に最も近いバイト単位の実行アドレスを求めるためである。また、アドレス演算ユニットAUでの加算結果の下位2ビットをマスクしているのは、対象となるビットフィールド全体もしくはその先頭部分を含む32ビットのワードのアドレスを得るためである。

第7のステップS07では、上記のようにして得られたレジスタAUO内のワードアドレスがアドレス・アウトプット・レジスタAORに転送されてI／Oインタフェースを介して外部へ出力されるとともに、I／Oインタフェースに対しては外部データバス上のデータのフェッチを要求する指令がなされる。これによって、求めるビットフィールドの先頭部分を含むワードのメモリからのフェッチが開始される。これと並行して、レジスタAUOに保持されているアドレスがバスBCを介してテンポラリレジスタDTE2に転送される。また、ファンクションブロックFBには、ステップS05で得られたビットフィールドのアドレス0からのビット位置を示す値Lが、レジスタAOTからバスBAを介して供給され、その結果がレジスタFBOに格納される。これによって、ステップS06で得られた求めるビットフィールドの先頭部分を含むワードアドレス（オフセットが31以下のときはベースアドレスと一致する）からのビットフィールドの先頭位置Off・（これも一つのオフセット値であり、以下2次オフセットと称する）がレジスタFBOに保持されることになる。

続いて、ステップS08では、上記レジスタFBO内の値Off・（2次オフセット）がバスBCを介してテンポラリレジスタDTE2に転送される。これとともに、演算論理ユニットALUに対して、バスBAおよびBBを介して、テンポラリレジスタDTE3内の値WD・（ビットフィールド幅の下位5ビット）とレジスタFBO内の値Off・が供給されて加算され、その結果がレジスタALUOに格納される。これとともに、レジスタCBSに対し、制御部の側から定数「33」が設定される。「33」なる数は1ワードのビット数「32」に「1」を加えた数である。また、I／Oインタフェースからの信号に基づいてフェ

ッチしたデータすなわち求めるビットフィールドの内容が確定しているか確認する。データが確認している場合、そのデータはデータ・インプットレジスタＤＩＢに取り込まれていることになる。

次のステップＳ０９では、Ｉ／Ｏインタフェースによりフェッチされた値がデータ・インプット・レジスタＤＩＲからバスＢＣを介してテンポラリレジスタＤＴＥ２に転送される。これと並行して、演算論理ユニットＡＬＵではレジスタＡＬＵＯの値（Ｏｆｆ＊+WD＊）からレジスタＣＢＳの値「３３」の減算が行なわれ、その結果がレジスタＡＬＵＯに格納される。ここで、この減算結果が「正」ならばビットフィールドが２つのワードにまたがっていることを意味し、「負」ならば１ワード内に納まっていることを意味する。

また、ステップＳ０９では、次のステップにおいてバレルシフタＢＳＦで行なわれるビットシフト処理のシフト方向とシフト量の指定を行なう。具体的にはバレルシフタ・カウンタＢＣＮＴに対して右方向シフトの指示が与えられるとともに、レジスタＤＴＥ３内のビットフィールド幅WD＊がシフト量として供給される。

そして、ステップＳ１１では、バレルシフタＢＳＦに対して、レジスタＢＳＦＯの値と「０」が入力され、指定された方向とシフト量に従ったシフト動作が実行され、結果がレジスタＢＦＳＯに格納される。レジスタＢＳＦＯ内に入っていたビットフィールドの内容がフィールド幅ＷＤ＊の分だけ右シフトされると、３２ビットのレジスタＢＳＦＯ内には第１１図に示すようにフェッチされたビットフィールドの内容が右端に寄った状態すなわちレジスタの下位側から順に詰まった所望の状態で格納されるようになる。

このようにして、得られたビットフィールドの内容が、次のステップＳ１２において、レジスタＢＳＰＯからバスＢＣを介して汎用レジスタの一つRｂに格納される。

さらに、ステップＳ０９において、Ｏｆｆ＊とＷＤ＊との和と定数「３３」との減算結果が「正」となって、ビットフィールドがバウンダリ渡りをバスＢＡを介してレジスタＦＢＯ内の値Ｏｆｆ＊が、シフト量としてバレルシフタ・カウンタＢＣＮＴに供給される。

そして、ステップＳ１０において、バレルシフタＢＳＦに対してテンポラリレジスタＤＴＥ２の値すなわちメモリからフェッチされたビットフィールドの内容がバスＢＢを介して供給されるとともに、定数入力機能によって０が入力されてバレルシフト・カウンタＢＣＮＴの指示に従ったシフトが実行され、その結果がレジスタＢＳＦＯに格納される。これによって、３２ビットのレジスタＢＳＦＯ内には、第８図に示すようにフェッチされたビットフィールドの内容が左端に寄った状態つまりレジスタの上位ビット側から順に詰まった状態で格納される。これと並列して、ステップＳ１０では、次のステップにおいてバレルシフタＢＳＦで行なわれるシフト動作の方向指示と、シフト量の指定が行なわれる。すなわち、バレルシフタ・カウンタＢＣＮＴに対し、右方向シフトの指示が与えられ、かつバスＢＡを介してテンポラリしていると判定された場合には、ステップＳ１２から再びステップＳ０８に戻って上記手順を繰り返すことにより、複数のワードにまたがっているビットフィールドのすべての内容が読み出される。

第１０図に、上記マイクロフローにおけるオフセット値Ｏｆｆおよびビットフィールド幅ＷＤと２次オフセットＯｆｆ＊および端数ＷＤ＊との関係を図示しておく。

なお、第８図のマイクロフローに従った制限なしビットフィールド命令の実行手順では、バウンダリ渡りの判定を、２次オフセットＯｆｆ＊とビットフィールドの端数ＷＤ＊との和のから数「３３」を引いた結果が「正」か「負」かで行なっている。本来、バウンダリ渡りの判定は１次オフセットＯｆｆとビットフィールド幅ＷＤとから判定すべきであり、そのようなマイクロフローを記述することも可能である。ただし、１次オフセットOffにビットフィールド幅ＷＤを加えてそれをベースアドレスＢＡＤから３２ビットずつ区切っていくことでバウンダリ渡りが生じているか否か判定し

た場合と、実施例のように2次オフセットOff.にビットフィールドの端数WD.を加え、それが数「32」を超えたか否かでバウンダリ渡りの判定を行なった場合とで全く同じ結果が得られることは、第7図からも明らかである。

以上本発明者によってなされた発明を実施例に基づき具体的に説明したが、本発明は上記実施例に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることはいうまでもない。例えば上記実施例ではビットフィールドのベースアドレスやオフセット、フィールド幅及び演算の種類をオペランドで与えるようにしているが、オペランドの代わりに実効アドレス部で与えるようにしてもよい。

また、オペランドで演算の種類を指定するようにした本発明に係る演算命令は、従来の固定された演算の代わりに設けてもよいが、併用して設けるようにしてもよい。

以上の説明では主として本発明者によってなされた発明をその背景となった利用分野であるマイクロプロセッサの命令形式に適用した場合について説明したが、この発明はそれに限定されるものでなく、計算機やミニコン等プログラム制御方式のデータ処理システム一般の命令形式に利用することができる。

〔発明の効果〕

本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記のとおりである。

すなわち、プログラムに柔軟性を持たせ、例えばグラフィック処理用のプログラムの開発が容易に行なえるようになる。

4. 図面の簡単な説明

第1図及び第2図は本発明の適用の対象となったビットフィールド間演算命令におけるビットフィールドの構成例を示す説明図、

第3図は本発明に係るビットフィールド命令を実行するマイクロプロセッサの構成例を示すブロック図、

第4図は、第3図に示す実行ユニットの内部構成を示すブロック図、

第5図，第6図(A)及び第6図(B)は第4図に示す演算論理ユニットAUU、

第7図は本発明が適用されたビットフィールド間演算命令の実行シーケンスを示すフローチャート、

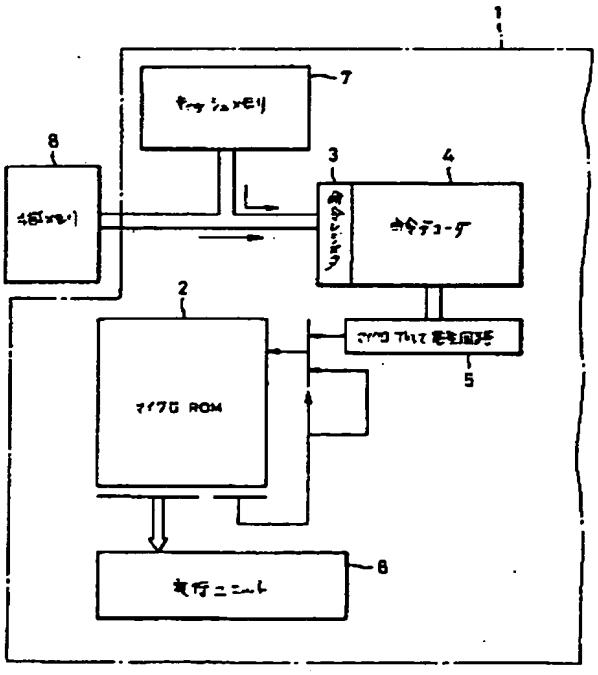
第8図は第7図におけるステップS1をさらに詳細に説明するための実行シーケンス、

第9図(A)，(B)はこの発明が適用される命令のフォーマットの実施例、

第10図及び第11図は、第8図に示す実行シーケンス中の動作の説明図である。

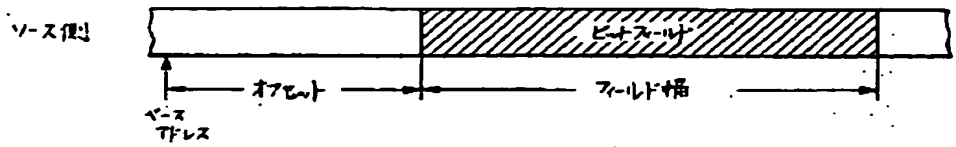
DOB…データ・アウトプット・レジスタ、DIR…データ・インプット・レジスタ、ALN…アライナ、BSF…バレルシフタ、BCNT…バレルシフタ・カウンタ、FB…ファンクションブロック、AU…アドレス演算ユニット、SFT…シフタ、AUO…ラッチ回路、AOR…アドレス・アウトプット・レジスタ、ALU…演算論理ユニット、INV…インバータ回路

第3図

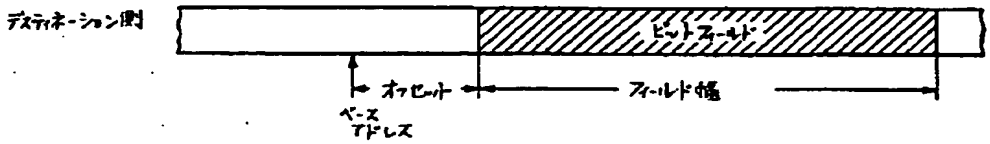


代理人 弁理士 小川勝男

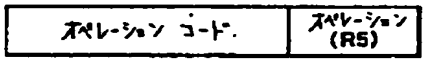
第 1 図



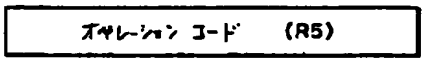
第 2 図



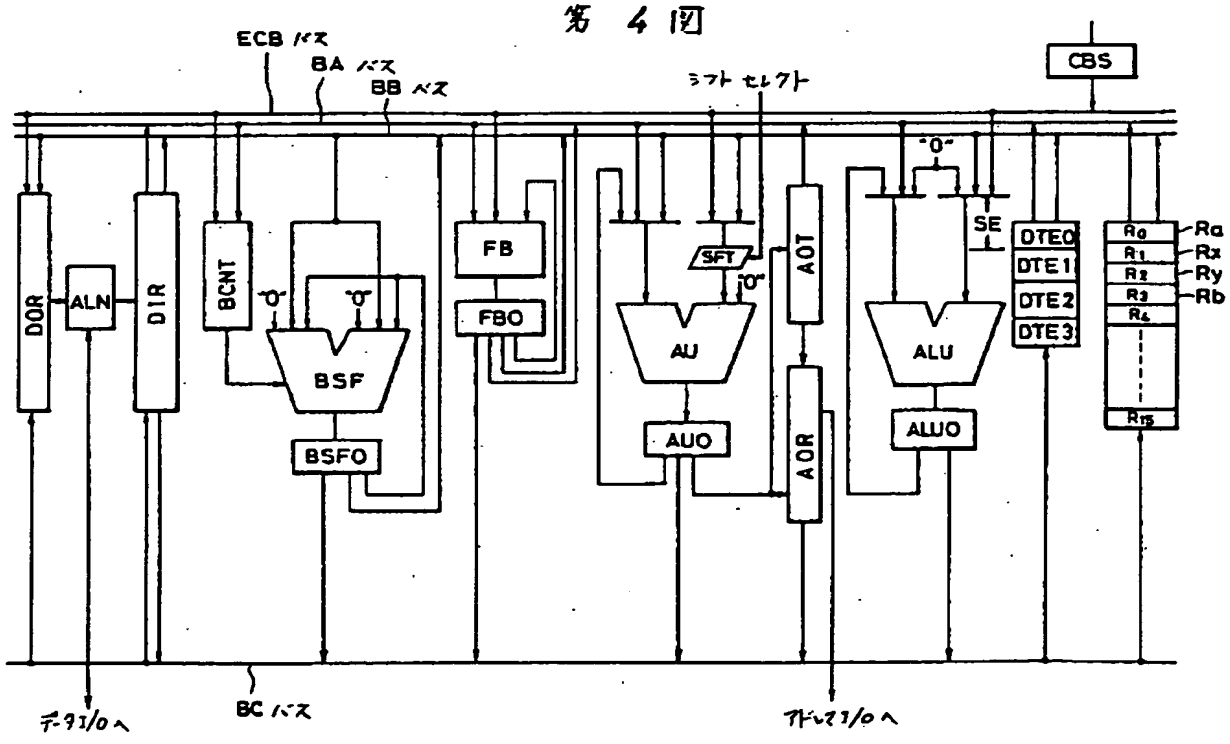
第 9(A) 図



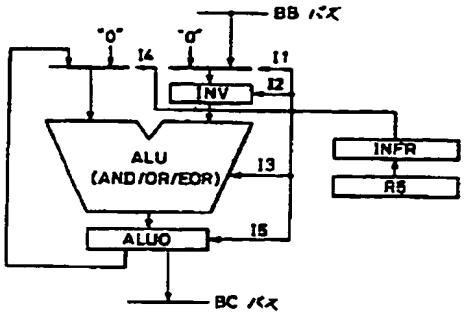
第 9(B) 図



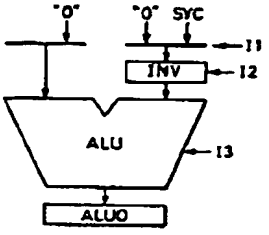
第 4 図



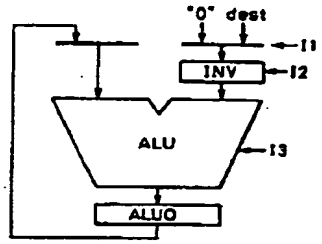
第5図



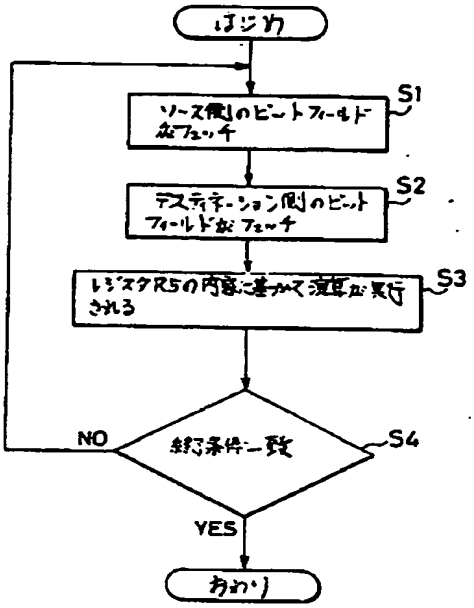
第6図(A)



第6図



第7図

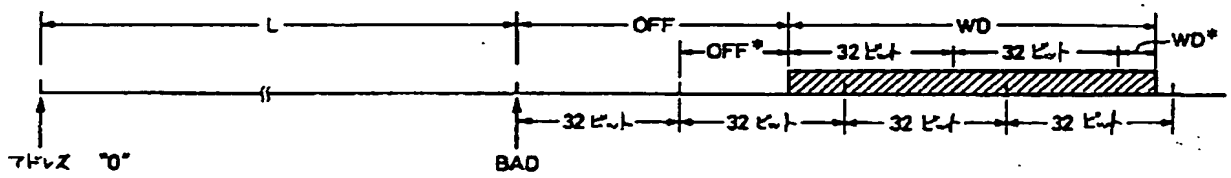


第8図

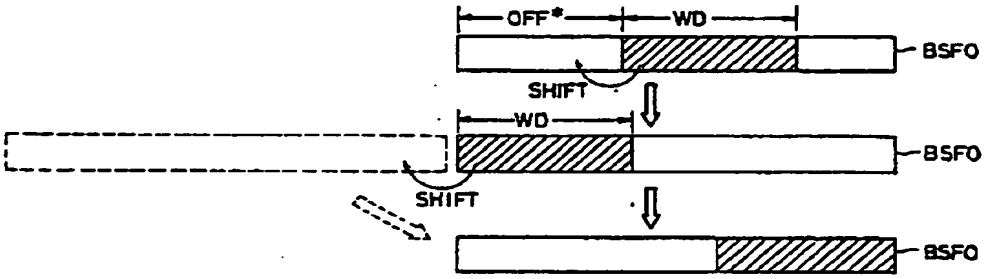
図面の説明

- ALUにBBの値とOを入力し、結果をALUOに格納する。
- ALUOの値をBCに転送し、同時にフェッチのリクエストをI/Oに対して指示する。
- I/Oにフェッチしたデータが確定しているかどうかを確認する。
- ALUでALUOとOを加算し、結果をALUOに格納する。
- ALUOの値をALUへ転送し、更にOTEDへBCバスを介して転送する。
- ALUにOとBBの値をBCバスを介して入力し、同時にこの値を符号拡張して加算し、結果をALUOに格納する。PBLにMの値をBCバスを介して入力し、ここで下位ビットだけを見るように上位をマスクし、結果をPBOに格納する。
- ALUOの値をBCバスを介してOTEIへ転送する。ALUでALUと同じBCバスを介して入力されるPBOの値を3ビット左シフトした値とを加算し、結果をALUOに格納する。
- ALUOの値をALUへ転送する。ALUでALUバスを介して入力されるOTEDの値とBBバスを介して入力されるOTEIの値を3ビット左シフトした値を加算し、この結果の下ビットマスクした値をALUOに格納する。
- PBOの値をBCバスを介してOTEIに転送する。
- ALUOの値をALUへ転送し、同時にフェッチのリクエストをI/Oに対して指令する。
- 更にALUOの値をBCバスを介してOTEIに転送する。
- PBLにMの値をBCバスを介して入力し、ここで下位ビットだけを見るように上位をマスクし、結果をPBOに格納する。
- PBOの値をBCバスを介してOTEIへ転送する。ALUでBCバスを介して入力されるOTEDの値とBBバスを介して入力されるPBOの値を加算し、結果をALUOに格納する。OBSへ定数33を設定する。I/Oにフェッチしたデータが確定しているかどうか確認する。
- メモリフェッチによって確定したDLIOの値をBCバスを介してOTEIへ転送する。ALUでALUOの値とBCバスを介して入力されるOBSの定数値とを減算して、結果をALUOへ格納する。BCINに右方向指示を示すと同時にBCバスを介してPBOの値を入力する。
- BSFにBCバスを介してOTEIOの値とIOを入力し、結果をBSFOへ格納する。BCINに右方向指示を示すと同時にBCバスを介してOTE30の値を入力する。
- BSFにOとBSFOの値を入力し、テストを行い、結果をBSFOに格納する。
- BCにBCバスを介してBSFOの値を格納する。

第 10 図



第 11 図



第1頁の続き

⑤Int.Cl.⁴	識別記号	庁内整理番号
G 06 F 15/72	400	6615-5B

⑦発明者 長谷川 淳 東京都小平市上水本町1479番地 日立マイクロコンピュータエンジニアリング株式会社内

⑦発明者 岩崎 一彦 東京都国分寺市東恋ケ窪1丁目280番地 株式会社日立製作所中央研究所内

手続補正書（方式）

昭和 63年 3月 11日

特許庁長官 殿

事件の表示

昭和 62 年 特許願 第 314063 号

発明の名称

マイクロプロセッサ

補正をする者

事件との関係 特許出願人

名 称 (510) 株式会社 日 立 製 作 所

(他 2 名)

代 理 人

居 所 〒100 東京都千代田区丸の内一丁目5番1号
株式会社日立製作所内
電話 東京 212-1111 (大代表)

氏 名 (6850) 弁理士 小 川 勝 男

補正命令の日付 昭和63年2月23日

補正の対象

図 面

補正の内容

明細書に添付した図面の第8図を別紙のとおり補正する。